A pro-Django presentation:
Brandon W. King

SoCal Piggies - May 2007

"How I created a non-wiki wiki in 5 days"

Non-wiki wiki?

- A website that looks like a website, but behaves like a wiki.
- Mediawiki, MoinMoin, look like wikis.
- Websites tend to have navigation menus.
- Wikis tend to have navigation within the wiki page.

Why?

- I wanted a web site for the lab that:
  - Looked like a website.
  - Had a menu manager.
  - And could be edited by biologists in the lab.

Requirements

- Has to be written faster than Diane can learn Zope.
- Allowing edits from command-line.
- Revision control of pages

The How:

Oswd.org

Django

PySVN

FCKEditor

Initialize Django

- Django-admin.py startproject woldcms
- woldcms/
  - __init__.py
  - manage.py
  - settings.py
  - urls.py

Manage.py is your friend

- cd woldcms/
- python manage.py startapp pages
- woldcms/pages/
  - __init__.py
  - models.py
  - views.py

Update woldcms/settings.py

- Update database settings:

```
DATABASE_ENGINE = 'sqlite3'           # 'postgresql_psycopg2', 'postgresql', 'mysql', 'sqlite3'
    or 'ado_mssql'.
DATABASE_NAME = '/home/king/proj/woldlab-2.0/woldlabwww.db'           # Or path to
    database file if using sqlite3.
DATABASE_USER = ''            # Not used with sqlite3.
DATABASE_PASSWORD = ''        # Not used with sqlite3.
DATABASE_HOST = ''            # Set to empty string for localhost. Not used with sqlite3.
DATABASE_PORT = ''            # Set to empty string for default. Not used with sqlite3.
```

- Update installed apps:

```
INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'woldcms.page'
)
```

Time to create the Models

- Django uses a custimized sqlalchemy for DB layer.
- On to the models…(After documentation detour)

The essential documentation

Make sure to read the following documentation. The rest (in the "Reference" section below) can be read in any particular order, as you need the various functionality.

- Django overview
- Installation guide
- Tutorial: Writing your first Django app
  - Part 1: Initialization, creating models, the database API
  - Part 2: Exploring the automatically-generated admin site
  - Part 3: Creating the public interface views
  - Part 4: Simple form processing and generic views
- Frequently asked questions (FAQ)
- How to read this documentation

Reference

- The django-admin.py and manage.py utilities
- Models: Creating models | Examples | The database API | Transactions
- Templates: Guide for HTML authors | Guide for Python programmers
- The newforms library | The old forms and manipulators library
- **New:** Testing Django applications
- Sessions
- Caching
- Internationalization
- Middleware
- Settings files
- URL configuration
- Request and response objects
- Generic views
- Authentication
- The django.contrib add-ons
  - **New:** Databrowse
  - Syndication feeds (RSS and Atom) (`django.contrib.syndication`)
  - Flatpages (`django.contrib.flatpages`)
  - Redirects (`django.contrib.redirects`)

Report a bug

Recent comments posted to djangoproject.com

Done

Add the models

- Page
  - MenuSet
    - Horizontal Menu (ref to “Menu”)
    - Vertical Menu (ref to “Menu”)
- Menu
  - Menu Item (Many-to-many)
    - Menu Sub Item (Many-to-many)

Page Model

```
from django.db import models

class MenuSubItem(models.Model):
  class Admin: pass

  # Internal name (i.e. mussa_download)
  id_name = models.CharField(maxlength=100)
  # Name to display on page
  display_name = models.CharField(maxlength=100)
  # Priority
  priority = models.PositiveSmallIntegerField(default=50)
  # Internal
  internal = models.BooleanField(default=False)

  # Link to internal page
  page_name = models.SlugField(maxlength=50, blank=True)
  # Link to url
  url = models.URLField(blank=True)

  def __str__(self):
    return '(%s) %s' % (self.id_name, self.display_name)
```

Menu Model

```
class Menu(models.Model):
  class Admin: pass

  name = models.CharField(maxlength=100)
  menu_items =
    models.ManyToManyField(MenuItem)

  def __str__(self):
    return self.name
```

Menu Item Model

```
class MenuItem(models.Model):
  class Admin: pass

  # Internal name (i.e. mussa_download)
  id_name = models.CharField(maxlength=100)
  # Name to display on page
  display_name = models.CharField(maxlength=100)
  # Priority
  priority = models.PositiveSmallIntegerField(default=50)
  # Internal
  internal = models.BooleanField(default=False)

  # Link to internal page
  page_name = models.SlugField(maxlength=50, blank=True)
  # Link to url
  url = models.URLField(blank=True)

  # Any sub-menu items (optional)
  sub_menu_items = models.ManyToManyField(MenuSubItem, blank=True)

  def __str__(self):
    return '(%s) %s' % (self.id_name, self.display_name)
```

Model Field Types

- AutoField
- BooleanField
- CharField
- CommaSeparatedIntegerField
- DateField
- DateTimeField
- DecimalField
- EmailField
- FileField
- FilePathField
- FloatField
- ImageField
- IntegerField
- IPAddressField
- NullBooleanField
- PhoneNumberField
- PositiveIntegerField
- PositiveSmallIntegerField
- SlugField
- SmallIntegerField
- TextField
- TimeField
- URLField
- USStateField
- XMLField

Model Field Options

- null
- blank
- choices
- core
- db_column
- db_index
- default
- editable
- help_text
- primary_key
- radio_admin
- unique
- unique_for_date
- unique_for_month
- unique_for_year
- validator_list

Create db, tables, and syncdb

- In woldcms/
  - python manage.py sqlall pages #creates new db (if d.n.e.), tables, indexes, etc.
  - python manage.py syncdb #Sets up authentication, admin, etc.

Fireup some Django goodness

- cd woldcms/
  - python manage.py runserver #now on localhost:8000
- <snap fingers> now at:
  - http://woldlab.caltech.edu/admin/

Mmm... data... Got Views?

- We can now input the data we need through the admin interface.

A simple view

```
from django.template import Context, loader
from django.http import HttpResponse,
    HttpResponseRedirect

def index(request):
  return HttpResponse("<h1>Look a non-
    wiki!</h1>")

def punish_user(request):
  return HttpRequestRedirect('http://microsoft.com')
```

The essential documentation

Make sure to read the following documentation. The rest (in the "Reference" section below) can be read in any particular order, as you need the various functionality.

- Django overview
- Installation guide
- Tutorial: Writing your first Django app
  - Part 1: Initialization, creating models, the database API
  - Part 2: Exploring the automatically-generated admin site
  - Part 3: Creating the public interface views
  - Part 4: Simple form processing and generic views
- Frequently asked questions (FAQ)
- How to read this documentation

Reference

- The django-admin.py and manage.py utilities
- Models: Creating models | Examples | The database API | Transactions
- Templates: Guide for HTML authors | Guide for Python programmers
- The newforms library | The old forms and manipulators library
- **New:** Testing Django applications
- Sessions
- Caching
- Internationalization
- Middleware
- Settings files
- URL configuration
- Request and response objects
- Generic views
- Authentication
- The django.contrib add-ons
  - **New:** Databrowse
  - Syndication feeds (RSS and Atom) (`django.contrib.syndication`)
  - Flatpages (`django.contrib.flatpages`)
  - Redirects (`django.contrib.redirects`)

Report a bug

Recent comments posted to djangoproject.com

Done

The ONLY down side to Django...

- Now to hook up the view to a url.
  - {% cue documentation %}

The essential documentation

Make sure to read the following documentation. The rest (in the "Reference" section below) can be read in any particular order, as you need the various functionality.

- Django overview
- Installation guide
- Tutorial: Writing your first Django app
  - Part 1: Initialization, creating models, the database API
  - Part 2: Exploring the automatically-generated admin site
  - Part 3: Creating the public interface views
  - Part 4: Simple form processing and generic views
- Frequently asked questions (FAQ)
- How to read this documentation

Reference

- The django-admin.py and manage.py utilities
- Models: Creating models | Examples | The database API | Transactions
- Templates: Guide for HTML authors | Guide for Python programmers
- The newforms library | The old forms and manipulators library
- **New:** Testing Django applications
- Sessions
- Caching
- Internationalization
- Middleware
- Settings files
- URL configuration
- Request and response objects
- Generic views
- Authentication
- The django.contrib add-ons
  - **New:** Databrowse
  - Syndication feeds (RSS and Atom) (`django.contrib.syndication`)
  - Flatpages (`django.contrib.flatpages`)
  - Redirects (`django.contrib.redirects`)

Report a bug

Recent comments posted to djangoproject.com

Done

- woldcms/urls.py:

```
from django.conf.urls.defaults import *
from woldcms import settings

urlpatterns = patterns('',
    # Uncomment this for admin:
    (r'^admin/', include('django.contrib.admin.urls')),
    (r'^html/', include('woldcms.page.urls')),
    #(r'^accounts/login/', 'django.contrib.auth.views.login'),
    (r'^login/', 'woldcms.page.views.login'),
    (r'^logout/', 'woldcms.page.views.logout'),
    (r'^img/(?P<path>.*)$', 'django.views.static.serve', {'document_root':
      settings.MEDIA_ROOT}),
)
```

- woldcms/pages/urls.py:

```
from django.conf.urls.defaults import *

urlpatterns = patterns('',
   (r'(?P<page_name>[a-zA-Z0-9_]+)/edit/$', 'woldcms.page.views.edit'),
   (r'(?P<page_name>[a-zA-Z0-9_]+)/log/$', 'woldcms.page.views.revision_log'),
   (r'(?P<page_name>[a-zA-Z0-9_]+)/(?P<revision>\d+)/$', 'woldcms.page.views.display'),
   (r'(?P<page_name>[a-zA-Z0-9_]+)/$', 'woldcms.page.views.display'),
   (r'', 'woldcms.page.views.index'),
)
```

```
def display(request, page_name, revision=None):
 """
 Display a particular page or offer a chance to create a new page.
 """
 try:
   p = Page.objects.get(page_name=page_name)
 except:
     msg = """
     <a href="/admin/page/page/add/?page_name=%s">Click here to create the %s page</a>
     """ % (page_name, page_name)
     return HttpResponse("Sorry, %s does not exist.<br /><br />%s" % (page_name,msg))

 if p.internal and not request.user.is_authenticated():
   return HttpResponseRedirect('/login/?next=%s' % (request.path))

 v_menu_item_iter = p.menu_set.menu_vertical.menu_items.iterator()
 h_menu_item_iter = p.menu_set.menu_horizonal.menu_items.iterator()
 content = wiki_file.loadPage(p.page_name, revision)
 if request.user.is_authenticated():
     content += """<br /><br /><a href="/html/%s/edit/">Edit</a> |
     <a href="/html/%s/log/">Log</a> | <a href="/admin/page/page/%s/">Page Settings</a>
     """ % (p.page_name, p.page_name, p.id)
 t = loader.get_template('woldlab.html')

 c = Context({
      'page': p,
      'v_menu_item_iter': v_menu_item_iter,
      'h_menu_item_iter': h_menu_item_iter,
      'user': request.user,
      'content': content,
      'request': request,
      'settings': settings
     })
 return HttpResponse(t.render(c))
```

Have view... now for some web 2.0, err, web 2.0esk looking pages.

- Enter “Django Templates”
  - {% cue template %}

Warning: Ugly code ahead

```
{% block v_menu %}
<h1>Menu</h1>
<p>
{% if v_menu_item_iter %}
  {% for menu_item in v_menu_item_iter %}
    {% if menu_item.internal and  not request.user.is_authenticated %}
    {% else %}
      <a class="nav
      {% if menu_item.page_name %}
        {% ifequal page.page_name menu_item.page_name %}
          active
        {% endifequal %}
        " href="/html/{{ menu_item.page_name }}"
      {% else %}
        " href="{{ menu_item.url }}"
      {% endif %}
      >{{ menu_item.display_name }}</a><span class="hide"> | </span>
    {% endif %}
    {% if menu_item.sub_menu_items.iterator %}
      {% for sub_item in menu_item.sub_menu_items.iterator %}
        {% if sub_item.internal and not request.user.is_authenticated %}
        {% else %}
          {% if sub_item.page_name %}
            <a class="nav sub" href="/html/{{ sub_item.page_name }}">{{ sub_item.display_name }}</a><span class="hide"> | </span>
          {% else %}
            <a class="nav sub" href="{{sub_item.url}}">{{ sub_item.display_name }}</a><span class="hide"> | </span>
          {% endif %}
        {% endif %}
      {% endfor %}
    {% endif %}
  {% endfor %}
{% endif %}
</p>
{% endblock %}
```

```
{% extends "woldlab.html" %}
{% load i18n %}

{% block title %}
        {% trans 'Page not found' %}
{% endblock %}

{% block h_menu %}
<ul>
  <li><a href="/html/">Home</a></li>
</ul>
{% endblock %}

{% block v_menu %}
<h1>Menu</h1>
<p>
<a class="nav" href="/html/">Home</a><span class="hide"> | </span>
</p>
{% endblock %}

{% block content %}

<h2>{% trans 'Page not found' %}</h2>

<p>{% trans "We're sorry, but the requested page could not be found." %}</p>

{% endblock %}
```

The essential documentation

Make sure to read the following documentation. The rest (in the "Reference" section below) can be read in any particular order, as you need the various functionality.

- Django overview
- Installation guide
- Tutorial: Writing your first Django app
  - Part 1: Initialization, creating models, the database API
  - Part 2: Exploring the automatically-generated admin site
  - Part 3: Creating the public interface views
  - Part 4: Simple form processing and generic views
- Frequently asked questions (FAQ)
- How to read this documentation

Reference

- The django-admin.py and manage.py utilities
- Models: Creating models | Examples | The database API | Transactions
- Templates: Guide for HTML authors | Guide for Python programmers
- The newforms library | The old forms and manipulators library
- **New:** Testing Django applications
- Sessions
- Caching
- Internationalization
- Middleware
- Settings files
- URL configuration
- Request and response objects
- Generic views
- Authentication
- The django.contrib add-ons
  - **New:** Databrowse
  - Syndication feeds (RSS and Atom) (`django.contrib.syndication`)
  - Flatpages (`django.contrib.flatpages`)
  - Redirects (`django.contrib.redirects`)

Report a bug

Recent comments posted to djangoproject.com

Done

Feature Request!

- Diane's attempt to slow me down so she could learn Zope before I wrote my non-wiki wiki.
- Diane now mysteriously volunteers to talk about Zope 3. =o)
- Brandon quickly runs out of the room.

PySVN

- Diane requested for a way to edit non-wiki wiki pages from the file system and allow for editing from the browser.

- Enter PySVN.

woldcms.pages.wiki_file

```
def savePage(page_name, content, user="Unknown"):
  """
  saves content to page_name
  """
  _validate(page_name)

  file_path = os.path.join(settings.WCMS_PAGES_DIR,
     page_name)

  if os.path.exists(file_path):
    _updateFile(page_name, content, file_path, user)
  else:
    _newFile(page_name, content, file_path)
```

woldcms.pages.wiki_file

```
import pysvn
from pysvn import wc_status_kind as  wc_status

def _newFile(page_name, content, file_path):
  f = open(file_path, 'w')
  f.write(content)
  f.close()

  client = pysvn.Client()
  client.add(file_path)
  client.checkin([file_path], "Inital version of page: %s" %
     (page_name))
```

woldcms.pages.wiki_file

```
def _updateFile(page_name, content, file_path, user="Unknown"):
  client = pysvn.Client()
  status = client.status(file_path)

  assert len(status) == 1
  status = status.pop()

  if status['text_status'] == wc_status.unversioned:
    client.add(file_path)
    client.checkin(file_path, "Found unversioned page: %s" % (page_name))
  elif status['text_status'] == wc_status.normal:
    pass
  else:
    msg = "Don't know how to handle status: %s" % (status['text_status'])
    raise pysvn.ClientError(msg)

  f = open(file_path, 'w')
  f.write(content)
  f.close()
```

def _updateFile() continued

```
status = client.status(file_path)
assert len(status) == 1
status = status.pop()

if status['text_status'] == wc_status.modified:
  client.checkin([file_path], "Updating %s page from
   wiki. User: %s" % (page_name, user))
elif status['text_status'] == wc_status.normal:
  pass
else:
  msg = "Don't know how to handle status: %s" %
   (status['text_status'])
  raise pysvn.ClientError(msg)
```

FCKEditor

- http://www.fckeditor.net/
- Live demo in lab… everyone has a Mac but me… Why does it have to break in Safari?
  - Safari Compatibility Effort
- GPL, LGPL and MPL licenses

Thickbox 2; I mean 3.

- http://jquery.com/demo/thickbox/
  - Was meant as a demo of jquery Javascript library.
  - MIT License

Oops

- Svn update on every view; ok for one user; bad for live demo with 10+ users.
- Safari FCKEditor blunder. (I need a Mac.)

Useful Django Stuff

```
@login_required
def edit(request, page_name):
    """
    Request to edit a page
    """
```

Django new style forms

```
from django import newforms as forms
class ContactForm(forms.Form):
    subject = forms.CharField(max_length=100)
    message = forms.CharField()
    sender = forms.EmailField()
    cc_myself = forms.BooleanField()

>>> data = {'subject': 'hello',
   ...              'message': 'Hi there',
   ...              'sender': 'foo@example.com',
   ...              'cc_myself': True}
>>> f = ContactForm(data)
>>> f.is_valid() True
```

The essential documentation

Make sure to read the following documentation. The rest (in the "Reference" section below) can be read in any particular order, as you need the various functionality.

- Django overview
- Installation guide
- Tutorial: Writing your first Django app
  - Part 1: Initialization, creating models, the database API
  - Part 2: Exploring the automatically-generated admin site
  - Part 3: Creating the public interface views
  - Part 4: Simple form processing and generic views
- Frequently asked questions (FAQ)
- How to read this documentation

Reference

- The django-admin.py and manage.py utilities
- Models: Creating models | Examples | The database API | Transactions
- Templates: Guide for HTML authors | Guide for Python programmers
- The newforms library | The old forms and manipulators library
- **New:** Testing Django applications
- Sessions
- Caching
- Internationalization
- Middleware
- Settings files
- URL configuration
- Request and response objects
- Generic views
- Authentication
- The django.contrib add-ons
  - **New:** Databrowse
  - Syndication feeds (RSS and Atom) (`django.contrib.syndication`)
  - Flatpages (`django.contrib.flatpages`)
  - Redirects (`django.contrib.redirects`)

Report a bug

Recent comments posted to djangoproject.com

Done

Sort menus... easy.

- Added priority field (integer; default 50)
- Added a meta class; I mean class named meta. Is this class a meta class? To be answered next SoCal Piggies meeting.

Menu Item Model

```
class MenuItem(models.Model):
  class Admin: pass

  class Meta:
    ordering = ['priority', 'display_name']

  # Internal name (i.e. mussa_download)
  id_name = models.CharField(maxlength=100)
  # Name to display on page
  display_name = models.CharField(maxlength=100)
  # Priority
  priority = models.PositiveSmallIntegerField(default=50)
```

Advice: skim ALL the documentation

- Django has a LOT of useful features
- Many are easy to use.
- If you skim ALL the docs after the basic tutorial, you will have an idea of where to look when you need an additional feature.